# Exercises: Razor Views & Layouts

Problems for exercises and homework for the [“C# MVC Frameworks - ASP.NET Core” course @ SoftUni](https://softuni.bg/trainings/2197/csharp-mvc-frameworks-asp-net-core-november-2018).

# CHUSHKA

**CHUSHKA** (**C**entral **H**ierarchically-**U**niversal **S**ales **H**ost **K**ickstarter **A**pplication) is a universal web application for selling products, like a web shop. You have been tasked to implement this application by the **Codex Input / Output** organization. There are several requirements you must follow in the implementation.

## Technological Requirements

## The application should be an ASP.NET Core Web app.

## Database Requirements

The **Database** of the **CHUSHKA** application needs to support **3 entities**:

### User

* Has an Username
* Has a Password
* Has a Full Name
* Has an Email
* Has an Role – can be one of the following values (“User”, “Admin”)

### Product

* Has an Id – a **GUID** string or **integer**.
* Has a Name
* Has a Price – think of the right data type here
* Has a Description
* Has a Type – can be one of the following values (“Food”, “Domestic”, “Health”, “Cosmetic”, “Other”)

### Order

* Has an Id – a **GUID** string or **integer**.
* Has a Product – a Product
* Has a Client – an User
* Has an Ordered On – a DateTime

Implement the entities with the **correct datatypes**. Think of the relationships between data models.

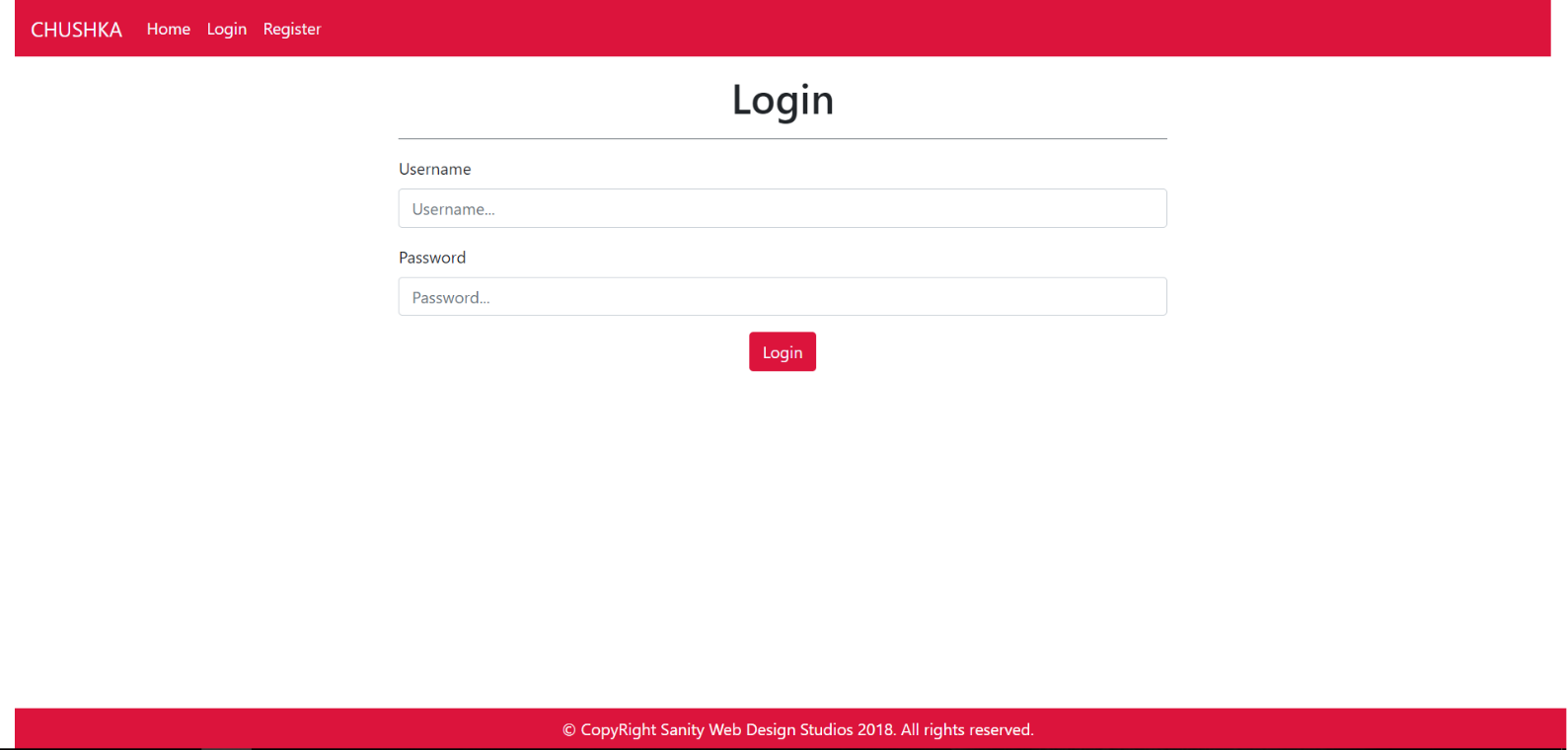
## Template Requirements

### Guest Templates

These are the **templates** and **functionalities**, accessible by Guests (**logged out** users).

#### Index Template

#### Login Template



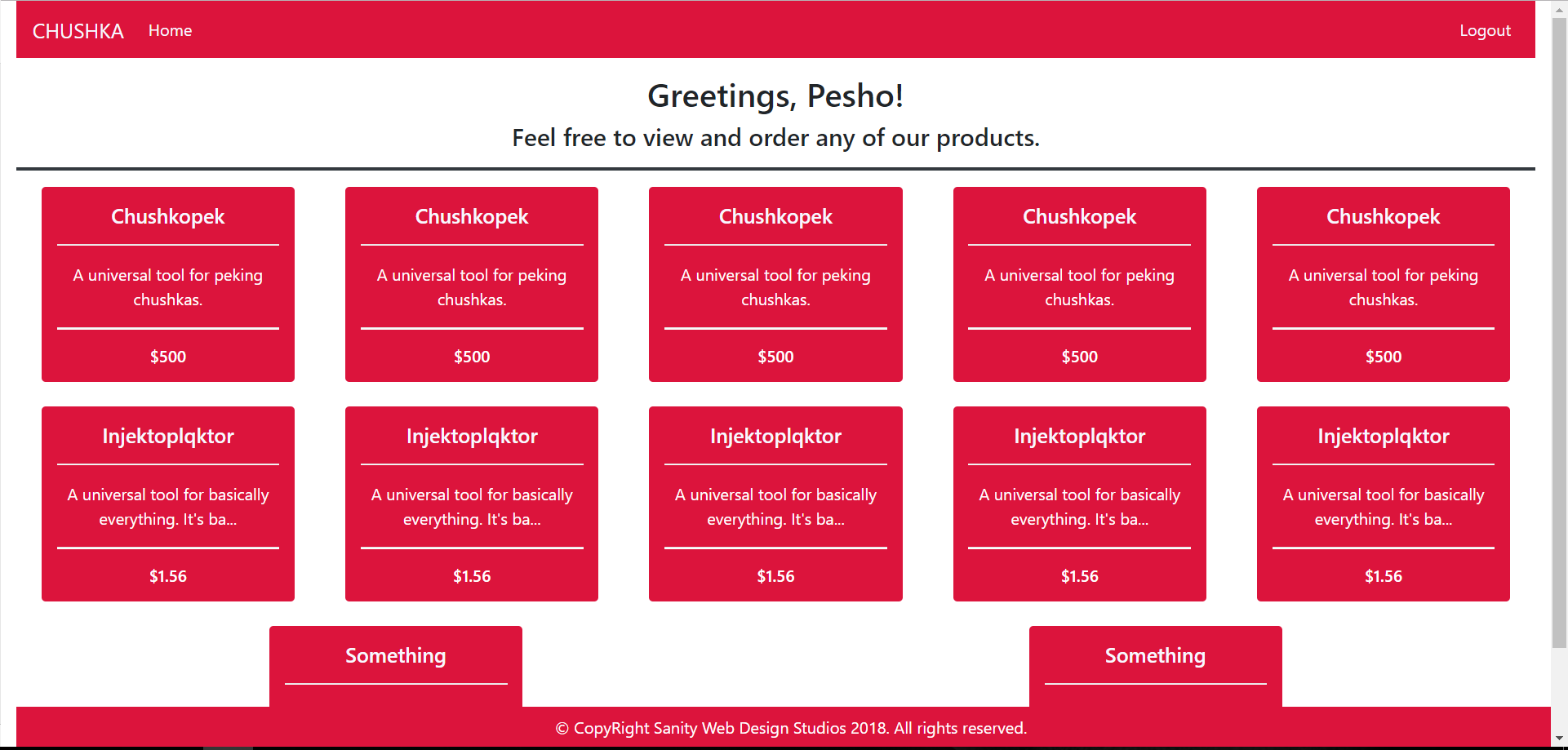
#### Register Template

### User Templates

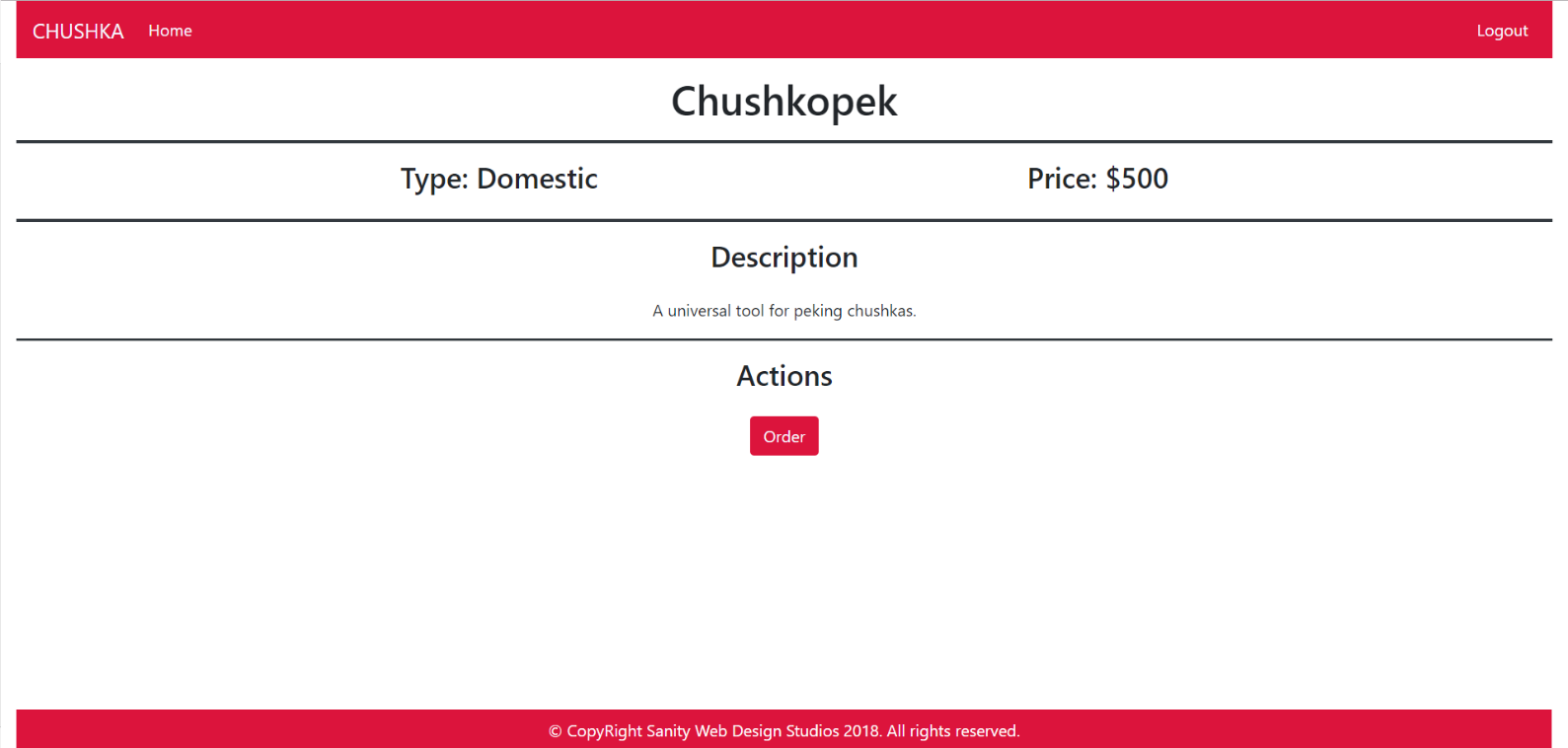
These are the **templates** and **functionalities**, accessible by Users (**logged in** users with Role - User).

#### Index Template (logged-in users)

**NOTE**: Products on this page have their description substringed to the 50th symbol, and then accompanied by 3 dots “…”



#### Product Details Template (logged-in users)

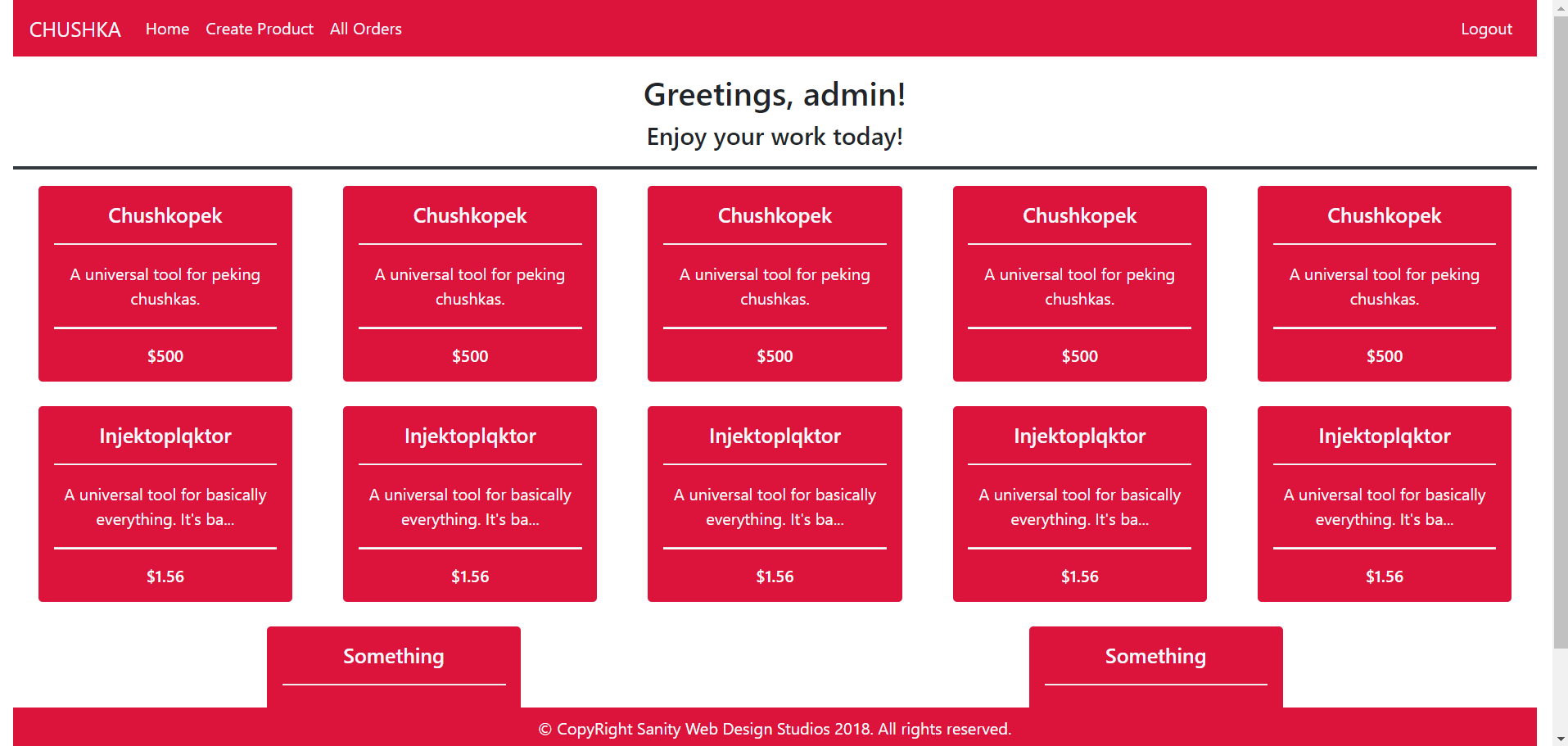


### Admin Templates

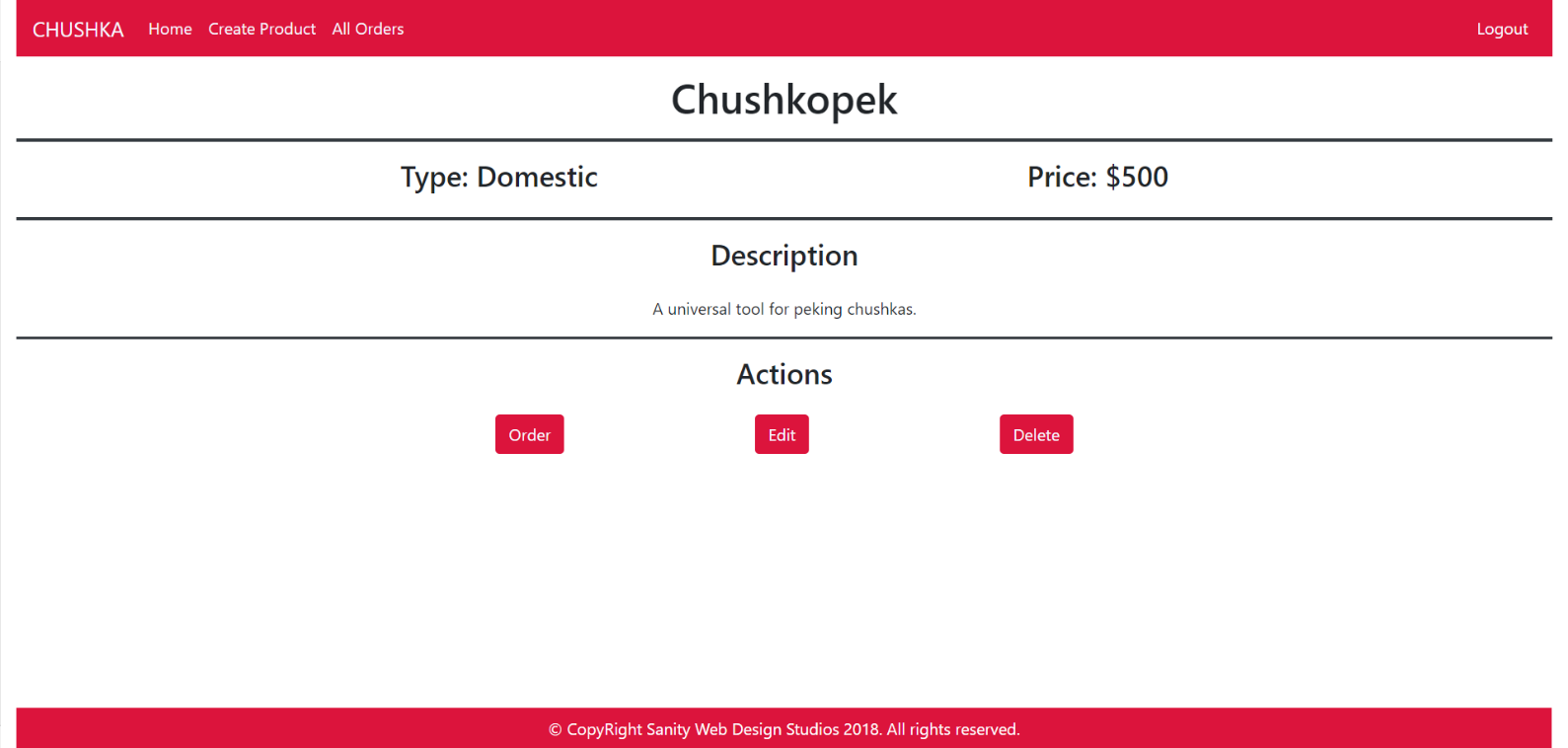
These are the **templates** and **functionalities**, accessible by Admins (**logged in** users with Role - Admin).

#### Index Template (logged-in admin)

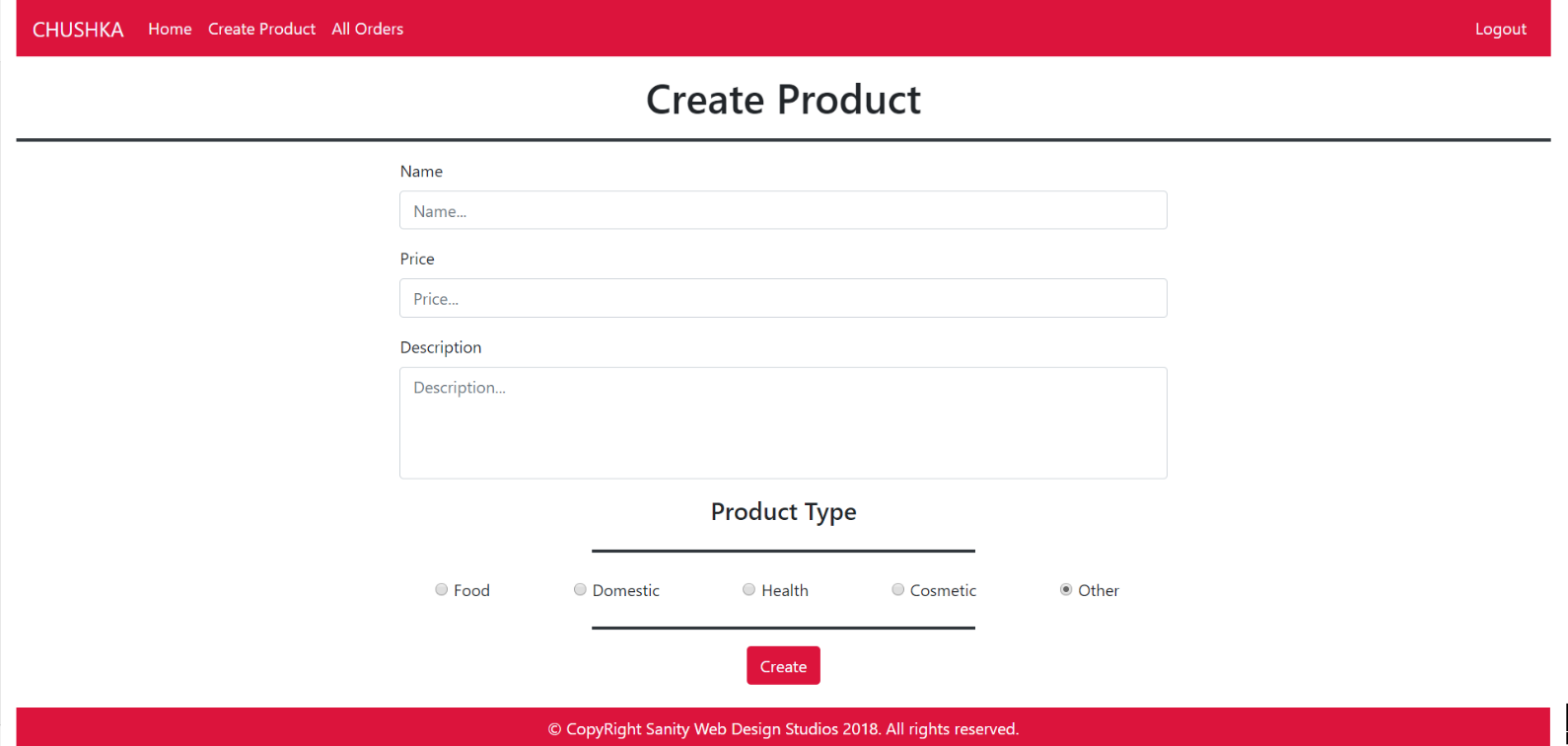
**NOTE**: Products on this page have their description substringed to the 50th symbol, and then accompanied by 3 dots “…”



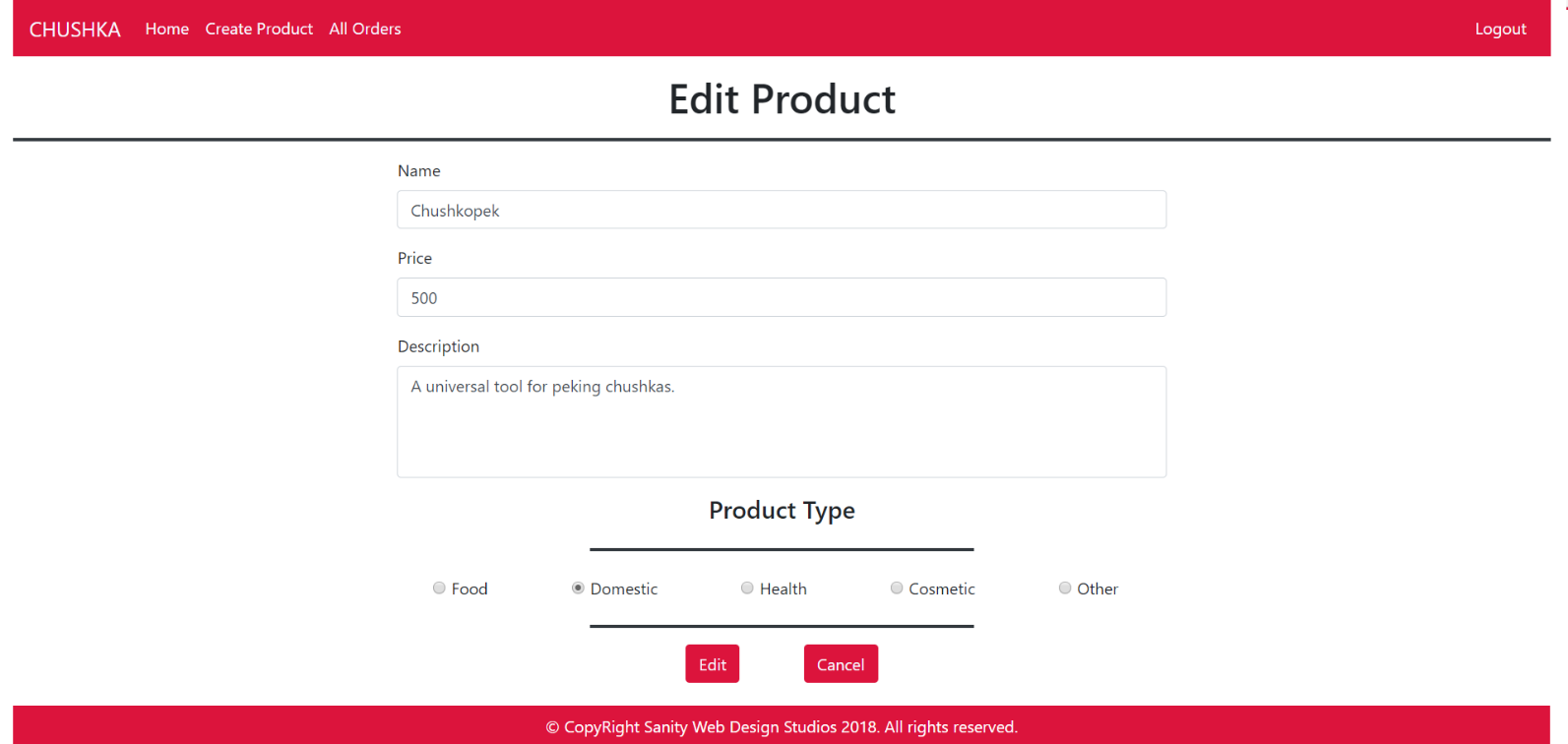
#### Product Details Admin Template (logged-in admin)



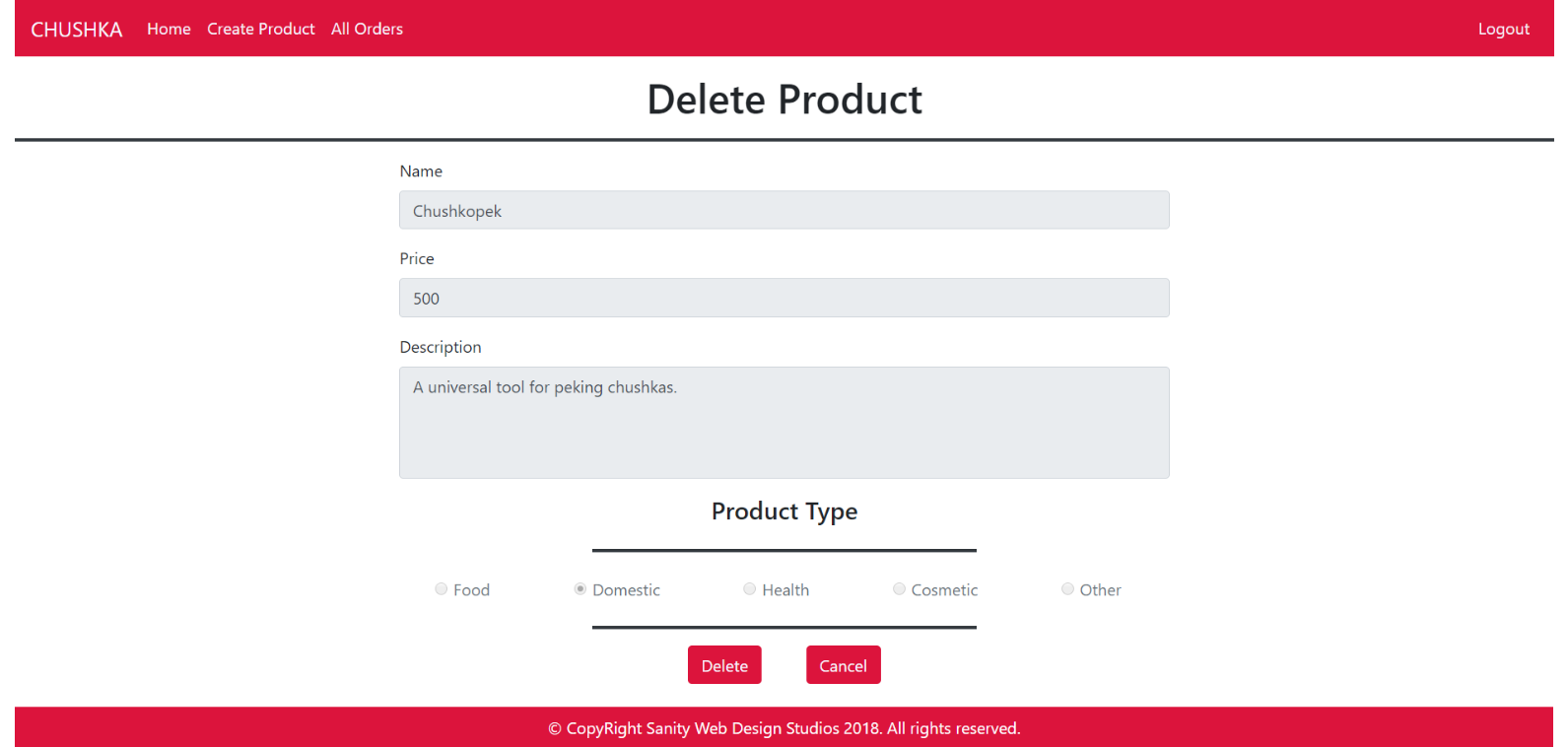
#### Product Create Template (logged-in admin)



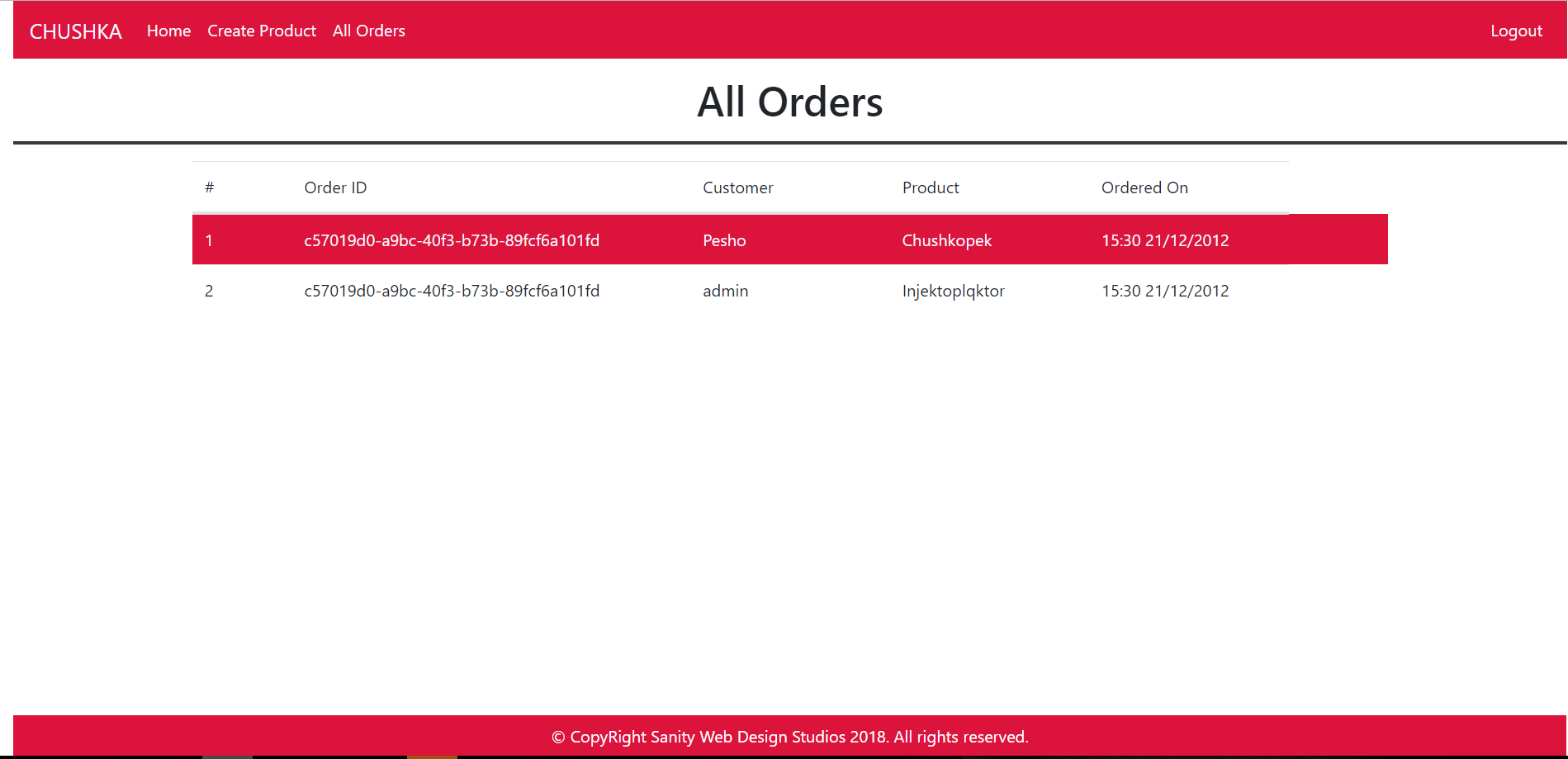
#### Product Edit Template (logged-in admin)



#### Product Delete Template (logged-in admin)



#### All Orders Template (logged-in admin)



Some of the templates have been given to you in the application skeleton, but the others will be for you to implement, so make sure you implement them correctly. You can use the given ones as helpers.

**NOTE**: The templates should look **EXACTLY** as shown above.

**NOTE**: The templates do **NOT** **require** **additional** **CSS**. Only **bootstrap** and the **given css** are enough.

## Functional Requirements

The Functionality Requirements describe the functionality that the **Application** must support.

The **application** should provide Guest (not logged-in) users with the functionality to login, register and **view** the Index page.

The **application** should provide Users (logged-in) with the functionality to logout, and view all products, and **order** a Product (clicking on the [Order] button on the Product Details page).

The **application** should provide Admins (logged-in) with the functionality to logout, and view all products, **order** a Product, **create** a Product, **edit** a Product, **delete** a Product, and **view** all Orders.

#### Users

When you register a new User, it should be assigned with a **role** – User.

Make the **first registered** user – an **Admin**.

#### Products

Upon clicking on the **rectangular object** holding the data about a certain product, you should be redirected to its Details page.

**NOTE**: Admins can Edit and Delete products on that page.

Upon **ordering** a Product, an Order should be **created** and **persisted**.

The **application** should **store** its **data** using Entity Framework Core.

## Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access Index page and functionality.
* Guest (not logged in) users can access Login page and functionality.
* Guest (not logged in) users can access Register page and functionality.
* Users (logged in) can access User LoggedIn Index page and functionality.
* Users (logged in) can access User Product Details page and functionality.
* Users (logged in) can access the Order Product functionality.
* Users (logged in) can access Logout functionality.
* Admins (logged in) can access **every functionality** a **normal** logged in User can.
* Admins (logged in) can access Admin LoggedIn Index page and functionality.
* Admins (logged in) can access the Admin Product Details page and functionality.
* Admins (logged in) can access the Product Create page and functionality.
* Admins (logged in) can access the Product Edit page and functionality.
* Admins (logged in) can access the Product Delete page and functionality.
* Admins (logged in) can access the All Orders page and functionality.

**TIP:** Add this code in Startup.cs to disable confirming email address of user:

services.Configure<IdentityOptions>(options =>

{

options.SignIn.RequireConfirmedEmail = false;

});

**TIP:** Add this code (remove the old one) in Startup.cs to enabling using Roles:

services.AddIdentity<IdentityUser, IdentityRole>()

.AddDefaultUI()

.AddDefaultTokenProviders()

.AddEntityFrameworkStores<EventuresDbContext>();

**TIP:** Change the code in **Register.cshtml.cs** to match the requirements and **add Role to newly created User**.